Parallel Implementation of the Max_Min Ant System for the Travelling Salesman Problem on GPU

Gaurav Bhardwaj
Department of Computer Science and Engineering
Maulana Azad National Institute of Technology
Bhopal, India

Manish Pandey
Department of Computer Science and Engineering
Maulana Azad National Institute of Technology
Bhopal, India

ABSTRACT
In this paper, we have proposed an approach to implement Ant colony optimization algorithm especially Max-Min Ant System for solving Travelling Salesman problem on GPU. GPUs are specially designed microprocessor for graphical operation and can be used for general purpose operations. ACO is a nature based inspired algorithm based on heuristics to find the solution for combinatorial optimization problems such as TSP. In this paper we have discussed many different programming issues of GPUs using OpenCL such synchronized memory access and barriers. We have used a partial solution for the stochastic probability function used in ACO for the tour construction to increase the speed-up. Thus with this implementation we are able to gain a speedup of 4.01x in CPU parallel and up to 11.29x speedup in GPU parallel.

1. INTRODUCTION
Travelling salesman problem is an NP-hard problem in a set of combinatorial optimization problem. In travelling salesman problem we have to found a Hamiltonian circuit having minimum total edge weight. TSP has various applications such as JOB Scheduling, DNA sequencing, designing and testing VLSI circuits, graph coloring, vehicle routing etc. There are various methods to solve such type problems such as ANT colony optimization, neural network, Genetic algorithm etc.

ACO is a heuristic algorithm for solving combinatorial optimization problem. ACO imitates the behavior of real ants to search food. Ants communicate indirectly to the agents of their colony with a trail of a chemical substance called pheromone. Pheromone is a chemical substance that shows the trace of an ant. Other ants follow the smell of the food and the trace of the pheromone to find out the minimum distance to the food.

Complex problem such as TSP needs huge computational power as well as time to solve. It takes lots of time for a single processor to solve such large problems single handedly. Parallel computing is the new paradigm to solve such type of problems using General Purpose Graphical Processing Unit. GPUs are meant to do graphical processing such as simple arithmetic operations also on graphics in the form of matrices. So we can utilize GPUs processor to solve our problem to speed up the computational time.

GPU consist of large no. of processors embedded together in a chip to perform a specific type of operations. Open CL (OPEN Computing Language) is the framework used to write programs that can be executed on heterogeneous platforms.

This paper applies ACO to the Travelling Salesman Problem in heterogeneous platform using OpenCL framework to achieve parallelism in ACO. We have compared the time taken in sequential as well as the parallel program used to solve this problem.

2. RELATED WORK
Travelling salesman problem is one of the oldest mathematical problems in history. Scientist had a great interest to solve such type of problem using different approaches. M.Dorigo and T.stizzle in 1992 [6] has designed an biological approach to solve such type of combinatorial optimization problem such as Travelling Salesman Problem called ACO. The first ACO algorithm was proposed by them called Ant System basic approach on ACO. Then many other algorithm were proposed based on it such Max-Min [13][15] approach, Ant colony System. All these approaches are successors of Ant system. M.Dorigo has given the basic parallel approach to solve ACO parallel as he has discussed the basic parallel behavior of ants in real life. M.Middendorf has given the approach for multi colony ant algorithm where many colonies of ants co-operate each other in finding the solution. He has also given the approach of transferring of information between colonies [16]. There after many parallel approaches has been delivered with the parallel strategies. This paper describes the parallel implementation of ACO on heterogeneous platform using OpenCL and comparing their parameters.

3. TRAVELLING SALESMAN PROBLEM
Travelling Salesman Problem represents a set of problem called combinatorial optimization problem. In TSP a salesman is given a map of cities and he has to visit all the cities exactly once and return back to the starting city with the minimum cost length tour of all the possible tour present in that map. Hence the total no. of possible tour in a graph with n vertices is (n-1)!.

There are various approaches to solve TSP. Classical approach to solve TSP are dynamic programming, branch and bound which uses heuristic and exact method and results into exact solution. But as we know TSP is an NP-hard problem so the time complexity of these algorithms are of exponential time. So they can solve the small problem in optimal time but as compared to the large problem time taken by these algorithms are quite high. So no classical approach can solve this type of problem in reasonable time as the size of the problem increases complexity increases exponentially.

So many alternate approaches are used to solve TSP which may not give you the exact solution but an optimal solution in reasonable time. Methods like nearest neighbor, spanning tree based on the greedy approach are efficiently used to solve such type of problems with small size. To overcome this

different other approaches based on natural and population techniques such as genetic algorithm, stimulated annealing, bee colony optimization, particle swarm optimization etc. are inspired from these techniques.

4. ANT COLONY OPTIMIZATION

ANT colony optimization technique introduced by Marco Dorigo in 1991 is based upon the real ant behavior in finding the shortest path between the nest and the food. They achieved this by indirect communication by a substance called pheromone which shows the trail of the ant. Ant uses heuristic information of its own knowledge the smell of the food and the decision of the path travelled by the other ants using the pheromone content on the path. The role of the pheromone is to guide other ants towards the food.

Ant has the capability of finding the food from their nest with the shortest path without having any visual clues. At a given point where there are more than one path to reach to their food then ants distribute themselves on different paths and the path and lay pheromone trace on that path and return with same path. Thus the path with minimum distance will acquire more pheromone as compared to other paths as the ants will return faster from that path comparative to the other path. So the new ants coming in the search of food will move with probability towards the path having higher pheromone content as compared to the path having lower pheromone content and in the end all the ants will move towards the same path with the minimum shortest path to their food. Now figure 1 shows the behavior of ants going from the upward direction will return early as compared to the ants going from the downward direction so the pheromone content in the upward direction is more as compared to the downward direction due to that in the end all the ants will start moving towards the upward direction which is the shortest path to their food.

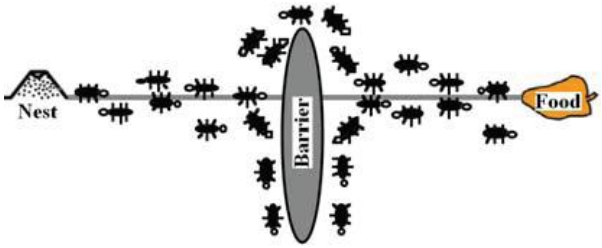

Fig 1: If necessary, the images can be extended both columns

ACO uses the set of artificial ants which co-operate each other to solve the problem and find the optimal solution of the problem. ACO can be used to solve combinatorial optimization problems such as Travelling Salesman Problem, Vehicle Routing, Quadratic Assignment, Graph Coloring, Project Scheduling, Multiple Knapsack etc. maximum of the problems are NP-hard problem i.e. they take exponential time complexity in their worst case.

In the travelling salesman problem we are given with a set of cities and the distance between them. We have to found a shortest tour such that each city should be visited exactly once and then return to the stating city. Formally we can say that we have to found a minimal Hamiltonian circuit in a fully connected graph.

In ACO we stimulate no. of artificial ants on a graph where each vertex represents the city and the edge represents the connection between the two cities. Pheromone is associated with each edge which shows the trace of the ant can be read and modified by the ants. It is an iterative algorithm where no. of ants is used to construct a solution from vertex to vertex without visiting any vertex more than once. At every vertex ant select the next vertex to be visited stochastically that is based upon the pheromone as well as the heuristic information available to it.

ACO algorithm
set parameters and pheromone value
    while termination condition not met do
    construct Ant solutions
    update pheromone
endwhile

in the above algorithm artificial ants will construct a solution. Ants start with an empty partial solution. At each iteration partial solution is modified by adding a set of components and updating the pheromone content. Creating a solution is completely based on a probabilistic stochastic mechanism. Updating pheromone value means increasing the pheromone content on the edges having good solution in order to find the optimal solution.

4.1 Max-Min Ant System

Max min ant system [13][15]is the successor to the main ACO based algorithm ant system. Many modification are made in MMAS with respect to AS. Firstly MMAS exploits only the best tour found in iteration or the best so far tour. That means the ant is allowed to update the pheromone content on the best so far tour or the best tour in iteration. This may lead to stagnation that is the ant may follow the same path in all iteration which may lead to the suboptimal solution because of the excessive pheromone deposition on the best tour till now which may not be optimal. To overcome this second modification is done to limit the maximum and minimum range of the pheromone content. The pheromone content lies between the range $[\tau_{min},\tau_{max}]$. Third modification is that the pheromone content is initialized with pheromone content equivalent to τ_{max} with a very low evaporation rate so that at initial level there is increase in exploration of paths. Fourth modification leads to reinitialized pheromone content if the system approaches the stagnation condition or there is no more exploration of edges till the time we are not able to get our optimal solution. With all such modifications MMAS is capable of exploring the path with more capability than AS and able to find the more optimal solution as compared to AS.

MMAS algorithm
Initialize pheromone
While termination condition met do
    If stagnation restart
    Construct ant solution
Update pheromone with the best tour or best so far
End while

4.1.1 Pheromone initialization

In MMAS pheromone is initially initialized equivalent to the value τ_{max} so that we can increase the exploration at initial level with a very low evaporation rate ρ. When an arc is not visited by any ant iteration than the pheromone is decreased by ρ where as next time it may leads to ρ^2 time the τ_{max} and so on which reduces the pheromone content. If incase we initialize the pheromone content with τ_{min} than the pheromone updating rate is quite high as compared to the evaporation rate which may lead to the initial stagnation.

4.1.2 Tour construction

In MMAS tour construction is done using a probabilistic stochastic mechanism using a probability function. An ant k

chooses next node j to be visited from node I with a probability P_{ij}. Such that

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{j\epsilon S_p}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & if\ j\epsilon S_p \\ 0 \end{cases}$$

Where S_p represents the set of cities not visited yet by ant k. probability of cities which are already visited is set to 0 so that they cannot be visited again. τ_{ij} is pheromone content on arc i to j. Where τ_{ij} is the pheromone content on the edge joining node i to j . η_{ij} represents the heuristic value which is inverse of the distance between the city i to j, which is given by:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

Where d_{ij} is the distance between the cities i to j. α and β represents the dependency of probability on the pheromone content or the heuristic value respectively. Increasing the value of α and β may vary the convergence of ACO.

4.1.3 Pheromone update

In MMAS pheromone content is updated only to the best tour in iteration or the best so far tour which may lead to good convergence point but it also lead to the stagnation as only some good arcs will have very high pheromone content where as bad arc will have very less pheromone rate as on each iteration the pheromone content decreases by ρ and after n iteration if the arc never came in the best solution may lead to the decreasing of pheromone content by ρ^n. Pheromone content is updated as:

$$\tau_{ij\leftarrow}\ \tau_{ij} + \Delta\tau_{ij}^{best}$$

Where $\Delta\tau\ \frac{best}{ij}$ = $1/C^{best}$ where C^{best} is the length of the best solution in the iteration or the best so far solution found in the iterations. In MMAS both best solution in iteration or the best so far solution is used in MMAS considering at what extend we are greedy about our solution. It has been studied that for small no. of cities iteration best is more considered to use but on other side with large no. of cities the best so far solution is considered. After iteration pheromone is evaporated with an evaporation rate ρ which is very low.

$$\tau_{ij} = \rho.\tau_{ij}$$

4.1.4 Pheromone limit

In MMAS as the name suggests has the upper and lower limit for the pheromone content $[\tau_{min},\tau_{max}]$. Pheromone content should be in this limit. If the pheromone content is more than that of τ_{max} after updating than it is set to τ_{max} similarly with the case of lower limit.

5. PARALLEL IMPLEMENTATION OF MMAS ON TSP

The main purpose of this section is to show parallel implementation of MMAS for TSP. Biologically ants use parallel approach in search of their food. Ants perform task based parallelism to search their food. All the ants search their food parallel simultaneously and synchronize with the help of the pheromone content in the ground similarly we can use this approach in artificial ants in ACO [12][14]. Parallel model used in ACO is a master/worker paradigm. Where master controls the workers by communicating and capturing the global knowledge where as worker implements the search. In this model same copies of the MMAS algorithm are simultaneously and randomly executed using different random source. ACO is an iterative approach where at each iteration master shares the global knowledge of pheromone to its worker ants to construct a solution. When 1000 of ants perform the search operation then the solution construction becomes comparatively fast as compared to sequential implementations. Parallelism where large number of threads can be executed simultaneously can be done using GPGPU (general purpose graphical processing unit). GPU [9] consist of hierarchy of processing elements and their memory. An AMD GPU consist of more than one SIMD (single instruction multiple data) computation engine. Where each computation engine consists of multiple thread processors which executes same instruction all the time simultaneously but data items may vary. Each thread processor has its own L1 cache. Each thread processor is a four or five way VLIW (very large instruction word) processor consisting of four or five ALUs respectively. Parallelism can be attained at both the level of thread processor and ALUs.

5.1 Pheromone initialize kernel

In this section we initialize the pheromone content using parallelism as every arc has to initialize a pheromone which is equal for every arc. Such type of task can be done using data parallelism approach using N*N size of work group. N is no. of cities in the problem.

Kernel Pheromone_initialize
for all (i,j) ϵ E in G
    Initialize τ_{ij}
End for

Where the size of the work group in this case is of N*N which initialize the pheromone matrix with some initialized value. τ_{ij} is the pheromone content at arc i to j in pheromone matrix τ. The value should be equivalent to τ_{max}. In parallel this kernel can be executed in O(1) as compared to the sequential $O(N^2)$.
Tour construction kernel

5.2 Tour Construction Kernel

In solution construction task based parallelism approach is used as each ant performs their task independent of each other to find the best tour. As discussed earlier in this phase ants are allocated the source node randomly and they have to visit each node exactly once and have to reach back to their source node. In iteration they have to choose their next node using probabilistic stochastic mechanism. This phase has inbuilt parallelism at the level of each ant as the biological ant find their tour. Tour is found using probabilistic function given in eq. 1 . now in order to calculate the P_{ij} we calculate F_{ij} as partial solution for $P_{ij.}$ Where

$$F_{ij} = \left[\tau_{ij}^{\alpha}\right]\left[\eta_{ij}^{\beta}\right]$$

Kenrel Partial_probability
for all (i,j) ϵ E in G
    set F(i,j)= $\left[\tau_{ij}^{\alpha}\right]\left[\eta_{ij}^{\beta}\right]$
end for

Maximum value of F_{ij} will retain the highest probability P_{ij} for selecting node j after visiting i. this can also be calculated using a work group size N*N. as it is all independent from each other. This partial solution can be calculated in a single instance in O(1).

Now instead of using complete probabilistic function we will use only partial solution F_{ij} to choose next city to visit. City having highest F_{ij} is choosing to visit if it's not visited. To check the visited city we use a flag table. This flag table reduces the complexity of checking whether the node has been visited or not from O(N) to O(1).

Flag table

0	0	0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7	8	9

In this flag table 0 represents that the node is not visited yet where as 1 represents the node has already ben visited. With the help of this flag table we can check whether the node has been visited or not.

Kernel Solution_Construction(d, F)// d as weight matrix
Source=get_global_id(0)
Length =0
Starting from the source till all the nodes visited do
For every unvisited node find j from i such that F(i,j)is max
Visit node j
Length = length + d_{ij}
Set flag(j)=1 //visited
Enter j into the tour
Enter source node into the tour
Length = length + $d_{j,\ source}$
//synchronization is done here
If global_length >= length
Set global_length = length
Id = get_global_id(0)
Barrier (CLK_GLOBAL_MEM_FENCE)
If id = get_global_id(0)
Update global_tour =tour

In this kernel we have done synchronization as there is a global object for length and id which contain the best solution in iteration or best so far and the id and no two kernels can access that objects simultaneously. Similarly we have used barrier to check whether all the kernels had updated their length than the tour can be updated to global memory object. Now at the host level there is a length and the tour for that length.

5.3 Kernel Pheromone update

In this kernel we have to update the kernel with the help of the eq-2. to update the pheromone we can use a kernel with a workgroup size of N where we can update the kernel with the best length tour available at global memory.

Pheromone update
j=get_global_id(0)
i= global_tour[j]
set $\tau_{i,\ i+1} = \tau_{i,\ i+1} + 1/global\_length$
if ($\tau_{i,\ i+1} > \tau_{max}$)
set $\tau_{i,\ i+1} = \tau_{max}$

This kernel only updates the pheromone with the tour having best solution so far or the best solution of the last iteration. This kernel can be executed in O(1).
Kernel pheromone evaporation

5.4 Kernel Pheromone evaporation

This kernel evaporates the kernel with the pheromone evaporation rate ρ as in eq. 3 this kernel is executed with a workgroup size of N*N so that N*N independent kernel can evaporate the kernel in a single instance.
Kernel Pheromone_Evaporation
for all (i,j) ϵ E in G
set $\tau_{ij} = \rho.\tau_{ij}$
end for

6. COMPARATIVE ANALYSIS AND PERFORMANCE EVALUATION

MMAS is implemented sequential as well as parallel to check the speedup of the algorithm to find the solution. Parameterization of the kernel to check the best solution is done. OpenCL parallel implementation on CPU and GPU are tested on the following hardware specifications:-:
AMD Radeon HD 6450(GPU): 2 Compute units, 625 MHz clock, 2048MB Global Mem., 32KB Local Mem., 256 work group size on a system having Intel Core i5 CPU 650 @ 3.2 GHz and 2048MB RAM with AMD APP SDK v2.8.
MMAS sequentially is implemented on the above given hardware specification with a randomly generated graph with different no. of nodes as well as some standard graphs to compare our results. Comparative analysis of the speed up of graph is shown with the sequential, CPU parallel and GPU parallel. In GPU parallel we have considered only the kernel execution time. All the kernels are executed parallel to each other with the control of the kernels are with Host CPU.
Fig 5 shows the speedup between sequential, CPU parallel and GPU parallel. In CPU parallel we have used OpenCL platform to rum the algorithm on CPU parallel. In GPU parallel same program is implemented on GPU. With respect to that we are able to achieve 4.01 times speed up in CPU and up to 11.29 times speed up in GPU.

7. CONCLUSION AND FUTURE WORK

In this paper we have proposed parallel implementation of the ant colony algorithm, especially MMAS, for the travelling salesman problem. We have implemented this on AMD Radeon HD 6450(GPU). We have considered many programming issues on OpenCL such as synchronization and barrier where as an approach to reduce the load of the tour construction process by some pre-processing of probability. We are able to attain the speed-up of 4.01 at CPU parallel and up to 11.29 in GPU. All the different steps are tried to be done at GPU in parallel only.

Future works include other ACO algorithm to be done parallel and more precise parallel tour construction kernel to form to increase the speed up. In addition to ACO an hybrid approach can be used to solve TSP and other combinatorial optimization problems.

Fig 1: Shows the speed up between sequential, CPU parallel and GPU parallel

8. REFERENCES

[1] E.Lawler, J.Lenstra, A.Kan, and D.Shomsys Wiley New York, 1987 The Travelling Salesman Problem

[2] M.Dorigo and T.Stizzle : Bradford Company 2004. Ant Colony Optimization.

[3] C.Blum. Physics of life reviews, vol. 2, no.4, pp. 353-373, 2005. Ant colony optimization: Introduction and recent trends.

[4] Y-S. You. Genetic and Evolutionary computation, 2009. Parallel ant system for Travelling Salesman Problem.

[5] K.D. boese , A.B. Kahng, and S.Muddu .Operations Research letters ,16:101-113,1994. A new adaptive multistart technique for combinatorial global optimization.

[6] M. Dorigo. PhD thesis, Politecnico di Milano, 1992. Optimization, Learning, and Natural Algorithms

[7] T. Stizzle and H. H. Hoos. Future Generation Computer Systems, vol. 16, no8, pp. 889–914, 2000. MAX–MIN ant system

[8] M. Dorigo and T. Stizzle, A Bradford Book,2004. Ant Colony Optimization.

[9] Ying Zhang. PHD Thesis 2006. Performance and power comparisons between fermi and cypress GPUs.

[10] A. Munshi, B. R. Gaster, T.G. Mattson, J. Fung, D. Ginsburg, Addison-Wesley pub., 2011. OpenCL Programming Guide.

[11]] G. Reinelt, ORSA Journal on Computing, vol. 3, pp. 376–384, 1991. Tsplib–a traveling salesman problem library.

[12] M. Manfrin, M. Birattari, T. Stizzle, and M. Dorigo. 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, vol. LNCS 4150. Springer-Verlag, 2006, pp. 224–234. Parallel ant colony optimization for the traveling salesman problem.

[13] T.Stizzle, H.Hoos MAX-MIN ant system and local search for the Travelling salesman problem

[14] A.Colorni, M.Dorigo, V.Manniezo. Distributed Optimization by ant colonies. ECAL-91 European conference of Artificial Life.

[15] T.Stizzle, H.Hoos. MAX-MIN Ant System.IRIDIA

[16] M.Midderndorf, F.rieschle, H.Schmeck . Multi Colony Ant Algorithms. Journal of heuristics, 8:305-320,2002